Speech and Language Processing

SLP Chapter 5

Today

- Parts of speech (POS)
- Tagsets
- POS Tagging
  - Rule-based tagging
  - HMMs and Viterbi algorithm

Parts of Speech

- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
  - Called: parts-of-speech, lexical categories, word classes, morphological classes, lexical tags...
  - Lots of debate within linguistics about the number, nature, and universality of these
    - We'll completely ignore this debate.

POS examples

- N noun *chair, bandwidth, pacing*
- V verb *study, debate, munch*
- ADJ adjective *purple, tall, ridiculous*
- ADV adverb *unfortunately, slowly*
- P preposition *of, by, to*
- PRO pronoun *I, me, mine*
- DET determiner *the, a, that, those*

POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a collection.

WORD	tag
the	**DET**
koala	**N**
put	**V**
the	**DET**
keys	**N**
on	**P**
the	**DET**
table	**N**

Why is POS Tagging Useful?

- First step of a vast number of practical tasks
- Speech synthesis
  - How to pronounce "lead"?
  - INsult inSULT
  - OBject obJECT
  - OVERflow overFLOW
  - DIScount disCOUNT
  - CONtent conTENT
- Parsing
  - Need to know if a word is an N or V before you can parse
- Information extraction
  - Finding names, relations, etc.
- Machine Translation

Open and Closed Classes

- Closed class: a small fixed membership
  - Prepositions: of, in, by, …
  - Auxiliaries: may, can, will had, been, …
  - Pronouns: I, you, she, mine, his, them, …
  - Usually function words (short common words which play a role in grammar)
- Open class: new ones can be created all the time
  - English has 4: Nouns, Verbs, Adjectives, Adverbs
  - Many languages have these 4, but not all!

Open Class Words

- Nouns /naamwoorden
  - Proper nouns (Boulder, Granby, Eli Manning)
    - English capitalizes these.
  - Common nouns (the rest).
  - Count nouns and mass nouns
    - Count: have plurals, get counted: goat/goats, one goat, two goats
    - Mass: don't get counted (snow, salt, communism) (*two snows)
- Adverbs / bijwoorden: tend to modify things
  - Unfortunately, John walked home extremely slowly yesterday
  - Directional/locative adverbs (here,home, downhill)
  - Degree adverbs (extremely, very, somewhat)
  - Manner adverbs (slowly, slinkily, delicately)
- Verbs / werkwoorden
  - In English, have morphological affixes (eat/eats/eaten)

Closed Class Words

Examples:

- prepositions / voorzetsels: *on, under, over, …*
- particles / partikels: *up, down, on, off, …*
- determiners /lidwoorden: *a, an, the, …*
- pronouns / voornaamwrd: *she, who, I, ..*
- conjunctions / voegwoorden: *and, but, or, …*
- auxiliary verbs / koppelww: *can, may should, …*
- Numerals /telwoorden: *one, two, three, third, …*

Prepositions from CELEX /voorzetsels

of	540,085	through	14,964	worth	1,563	pace	12
in	331,235	after	13,670	toward	1,390	nigh	9
for	142,421	between	13,275	plus	750	re	4
to	125,691	under	9,525	till	686	mid	3
with	124,965	per	6,515	amongst	525	o'er	2
on	109,129	among	5,090	via	351	but	0
at	100,169	within	5,030	amid	222	ere	0
by	77,794	towards	4,700	underneath	164	less	0
from	74,843	above	3,056	versus	113	midst	0
about	38,428	near	2,026	amidst	67	o'	0
than	20,210	off	1,695	sans	20	thru	0
over	18,071	past	1,575	circa	14	vice	0

English Particles / partikels

aboard	aside	besides	forward(s)	opposite	through
about	astray	between	home	out	throughout
above	away	beyond	in	outside	together
across	back	by	inside	over	under
ahead	before	close	instead	overhead	underneath
alongside	behind	down	near	past	up
apart	below	east, etc.	off	round	within
around	beneath	eastward(s),etc.	on	since	without

Conjunctions/voegwoorden

and	514,946	yet	5,040	considering	174	forasmuch as	0
that	134,773	since	4,843	lest	131	however	0
but	96,889	where	3,952	albeit	104	immediately	0
or	76,563	nor	3,078	providing	96	in as far as	0
as	54,608	once	2,826	whereupon	85	in so far as	0
if	53,917	unless	2,205	seeing	63	inasmuch as	0
when	37,975	why	1,333	directly	26	insomuch as	0
because	23,626	now	1,290	ere	12	insomuch that	0
so	12,933	neither	1,120	notwithstanding	3	like	0
before	10,720	whenever	913	according as	0	neither nor	0
though	10,329	whereas	867	as if	0	now that	0
than	9,511	except	864	as long as	0	only	0
while	8,144	till	686	as though	0	provided that	0
after	7,042	provided	594	both and	0	providing that	0
whether	5,978	whilst	351	but that	0	seeing as	0
for	5,935	suppose	281	but then	0	seeing as how	0
although	5,424	cos	188	but then again	0	seeing that	0
until	5,072	supposing	185	either or	0	without	0

POS Tagging
Choosing a Tagset

- There are so many parts of speech, potential distinctions we can draw
- To do POS tagging, we need to choose a standard set of tags to work with
- Could pick very coarse tagsets
  - N, V, Adj, Adv [synthesetaak!]
- More commonly used set is finer grained, the "Penn TreeBank tagset", 45 tags
  - PRP$, WRB, WP$, VBG
- Even more fine-grained tagsets exist

Penn TreeBank POS Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two, three*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb, base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb, 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	dollar sign	$
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Using the Penn Tagset

- The/DT (determiner) grand/JJ (adjective) jury/NN (singular noun) commmented/VBD (verb, past tense) on/IN (preposition) a/DT (determiner) number/NN (singular noun) of/IN (preposition) other/JJ (adjective) topics/NNS (plural noun) ./.

POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ (bijvoeglijk naamwoord)
  - On my *back* = NN (zelfstandig naamwoord)
  - Win the voters *back* = RB (bijwoord)
  - Promised to *back* the bill = VB (werkwoord)
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

These examples from Dekang Lin

How Hard is POS Tagging? Measuring Ambiguity

		87-tag Original Brown		45-tag Treebank Brown	
Unambiguous (1 tag)		**44,019**		**38,857**	
Ambiguous (2–7 tags)		**5,490**		**8844**	
Details:	2 tags	4,967		6,731	
	3 tags	411		1621	
	4 tags	91		357	
	5 tags	17		90	
	6 tags	2	(*well, beat*)	32	
	7 tags	2	(*still, down*)	6	(*well, set, round, open, fit, down*)
	8 tags			4	(*'s, half, back, a*)
	9 tags			3	(*that, more, in*)

Two Methods for POS Tagging

1. Rule-based tagging
   - (ENGTWOL)
     morfologische analyser/tagger: http://www2.lingsoft.fi/cgi-bin/engtwol?
2. Stochastic
   1. Probabilistic sequence models
      - HMM (Hidden Markov Model) tagging
      - MEMMs (Maximum Entropy Markov Models)

Rule-Based Tagging

- Start with a dictionary
- Assign all possible tags to words from the dictionary
- Write rules by hand to selectively remove tags
- Leaving the correct tag for each word.

Start With a Dictionary

- she: PRP (personal pronoun)
- promised: VBN (verb, past participle),VBD (verb, past tense)
- to TO (to)
- back: VB (verb), JJ (adjective), RB (adverb), NN (noun)
- the: DT (deteminer)
- bill: NN (noun), VB (verb)

- Etc… for the ~100,000 words of English with more than 1 tag

Assign Every Possible Tag

			NN		
			RB		
	VBN		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	**promised**	**to**	**back**	**the**	**bill**

Write Rules to Eliminate Tags

Eliminate VBN if VBD is an option when VBN|VBD follows "<start> PRP"

			NN			
			RB			
	VBN		JJ			VB
PRP	VBD	TO	VB	DT		NN
She	**promised**	**to**	**back**	**the**		**bill**

Stage 1 of ENGTWOL Tagging

- First Stage: Run words through FST morphological analyzer to get all parts of speech.
- Example: *Pavlov had shown that salivation …*

Pavlov	**PAVLOV N NOM SG PROPER**
had	**HAVE V PAST VFIN SVO**
	HAVE PCP2 SVO
shown	**SHOW PCP2 SVOO SVO SV**
that	ADV
	PRON DEM SG
	DET CENTRAL DEM SG
	CS
salivation	**N NOM SG**

Stage 2 of ENGTWOL Tagging

- Second Stage: Apply NEGATIVE constraints.
- Example: Adverbial "that" rule
  - Eliminates all readings of "that" except the one in
    - "It isn't *that* odd"

Given input: "that"

If
(+1 A/ADV/QUANT) **;if next word is adj/adv/quantifier**
(+2 SENT-LIM) **;following which is E-O-S** (end-of-sentence)
(NOT -1 SVOC/A) **; and the previous word is not a**
; verb like "consider" which
; allows adjective complements
; in "I consider that odd"

Then eliminate non-ADV tags
Else eliminate ADV

Hidden Markov Model Tagging

POS Tagging as Sequence Classification

- We are given a sentence (an "observation" or "sequence of observations")
  - *Secretariat is expected to race tomorrow*
- What is the **best sequence of tags** that corresponds to **this sequence of observations**?

POS Tagging as Sequence Classification

- Probabilistic view:
  - Consider *all possible sequences* of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \ldots w_n$.

Probleem

- Gegeven een woordreeks: wat is de meest waarschijnlijke POS sequentie?
- Gegeven een akoestisch signaal: wat is de meest waarschijnlijke foneemreeks?
- Gegeven een foneemreeks: wat is de meest waarschijnlijke woordenreeks?
- Gegeven een fotocopie van een tekst: wat is de meest waarschijnlijke woordenreeks?

"noisy channel", de waarneming bevat verstoringen (ambiguiteiten)

Hidden Markov Model Tagging

- Using an HMM to do POS tagging is a special case of *Bayesian inference*
  - Foundational work in computational linguistics
  - Bledsoe 1959: OCR
  - Mosteller and Wallace 1964: authorship identification
- It is also related to the "noisy channel" model that's the basis for ASR, OCR and MT
  automatic speech recognition, optical character reader, machine translation

Getting to HMMs

- We want, out of all sequences of n **tags $t_1 \ldots t_n$** the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ is highest.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- Hat ^ means "our estimate of the best one"
- Argmax_x f(x) means "the x such that f(x) is maximized"
- P(t|w) betekent de kans op een tag, gegeven een woord

Getting to HMMs

- This equation is guaranteed to give us the best tag sequence

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- But how to make it operational? How to compute this value?
- Intuition of Bayesian classification:
  - Use Bayes rule to transform this equation into a set of other probabilities that are easier to compute

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

- conditionele kansen
- POS: kans op tag, gegeven woord -> kans op woord gegeven tag
- ASH: kans op woord, gegeven spraak -> kans op spraakrealisatie, gegeven woord
- >> dat laatste is trainbaar!

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

$P(w_{1,n})$ is per definitie 1,
omdat de reeks waarvoor we de beste tagging zoeken, steeds dezelfde is

Using Bayes Rule

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n | t_1^n) P(t_1^n)$$

Likelihood and Prior

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

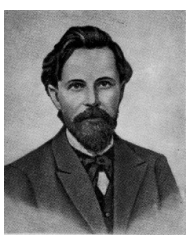
"Likelihood" van (kans op) de woordreeks, gegeven de tagreeks

"Prior probability" van de tagreeks (algemene kans op voorkomen van tagreeks)

Vereenvoudiging

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

Kans op woord hangt alleen van de bijbehorende tag af (is onafhankelijk van de omliggende tags)



$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Markoviaanse aanname: alleen de voorgaande doet mee (transitie waarschijnlijkheden)

samen

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

Te berekenen:

- Word likelihood probabilities

- Tag transition probabilities

stapsgewijs

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1})$$

- i=1:
  $P(t_1|t_0) = P(t_1)$ (per definitie), dan
  $P(t_1)$ (kans op tag1) *
  $P(w_1|t_1)$ (kans op woord 1 gegeven tag1)
- i=2:
  $P(t_2|t_1)$ (kans op overgang van tag1 naar tag2) *
  $P(w_2|t_2)$ (kans op woord 2 gegeven tag2)
- etc
  (doen voor alle mogelijke tagreeksen, en dan de tagreeks kiezen die de hoogste waarschijnlijkheid heeft)

Two Kinds of Probabilities

- Tag transition probabilities $p(t_i|t_{i-1})$
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect P(NN|DT) and P(JJ|DT) to be high
    - But P(DT|JJ) to be:

Tag transition probabilities

- Compute P($t_i \mid t_{i-1}$) by counting in a **labeled** corpus (trainingscorpus):

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

Tel de paren Determiner+Noun, en tel alle Determiners (ongeacht of er een Noun op volgt of niet)

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Word likelihood probabilities

- $p(w_i|t_i)$
  - VBZ (3sg Pres verb) likely to be "is"
  - Compute P($w_i|t_i$) by counting in a **tagged** corpus (training corpus):

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

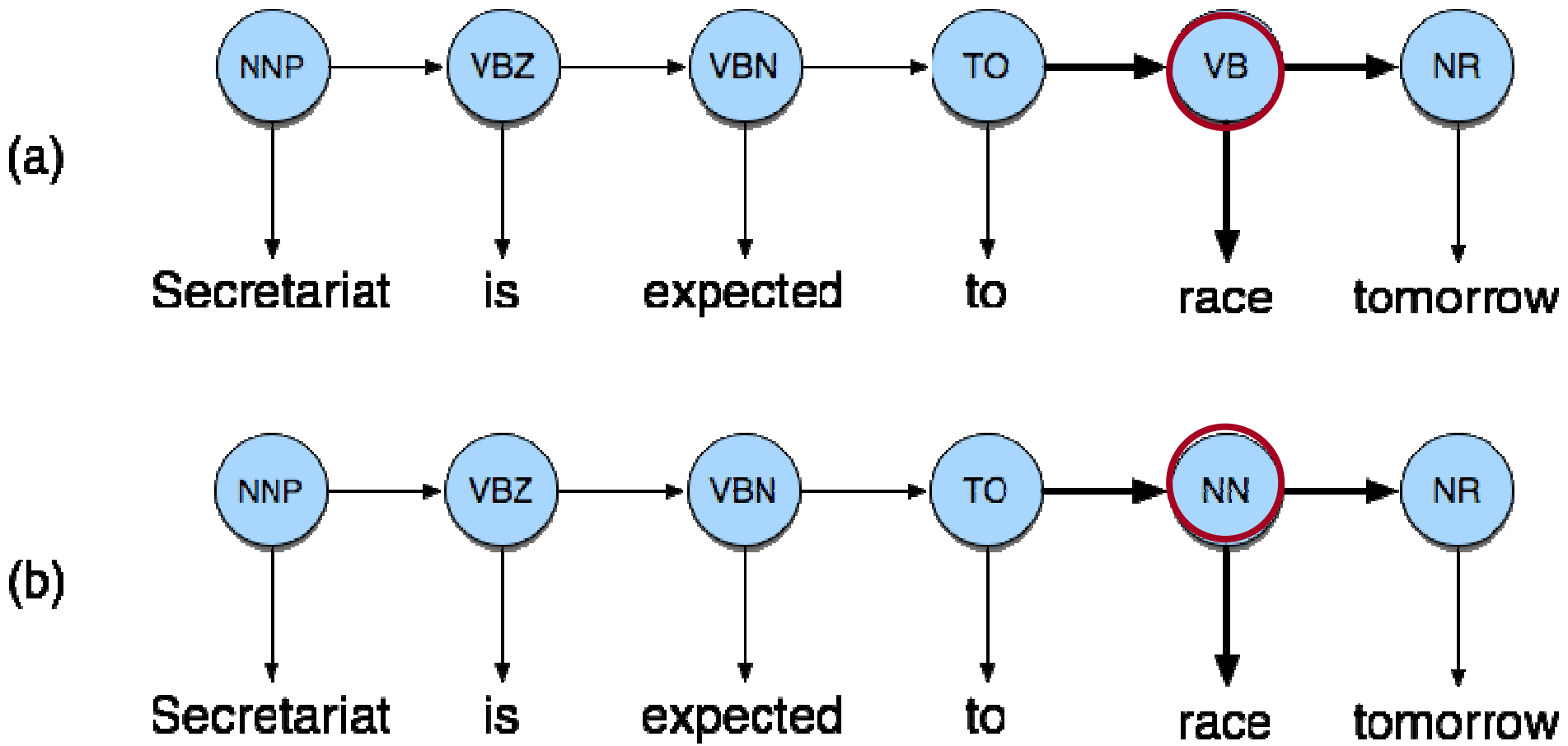
$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

Example: The Verb "race"

- Secretariat/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR (verb example)
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN (noun example)

- How do we pick the right tag?

Disambiguating "race"



Example

- P(NN|TO) = .00047 (NN is noun)
- P(VB|TO) = .83 (VB is verb)
- P(race|NN) = .00057
- P(race|VB) = .00012
- P(NR|VB) = .0027 (NR is adverbial noun)
- P(NR|NN) = .0012
- P(VB|TO)P(NR|VB)P(race|VB) = .00000027
- P(NN|TO)P(NR|NN)P(race|NN)=.00000000032
- So we (correctly) choose the verb reading
  [example only computes the differences in interpretation!]

to race

tomorrow

Hidden Markov Models

- What we've described with these two kinds of probabilities is a Hidden Markov Model (HMM)

  > POS: de tag die het woord produceerde, is 'verborgen'

  > ASH: het woord dat de spraak genereerde, is 'verborgen'

Hidden Markov Models

- Van *tag* naar *state*
- Een *state* is een instantie die output kan genereren

- Tag-state produceert woorden
- Phoneme-state produceert spraakgeluid

- Wie *observeren* woorden, maar weten niet precies welke tags die produceerden; die tags zijn *verborgen*

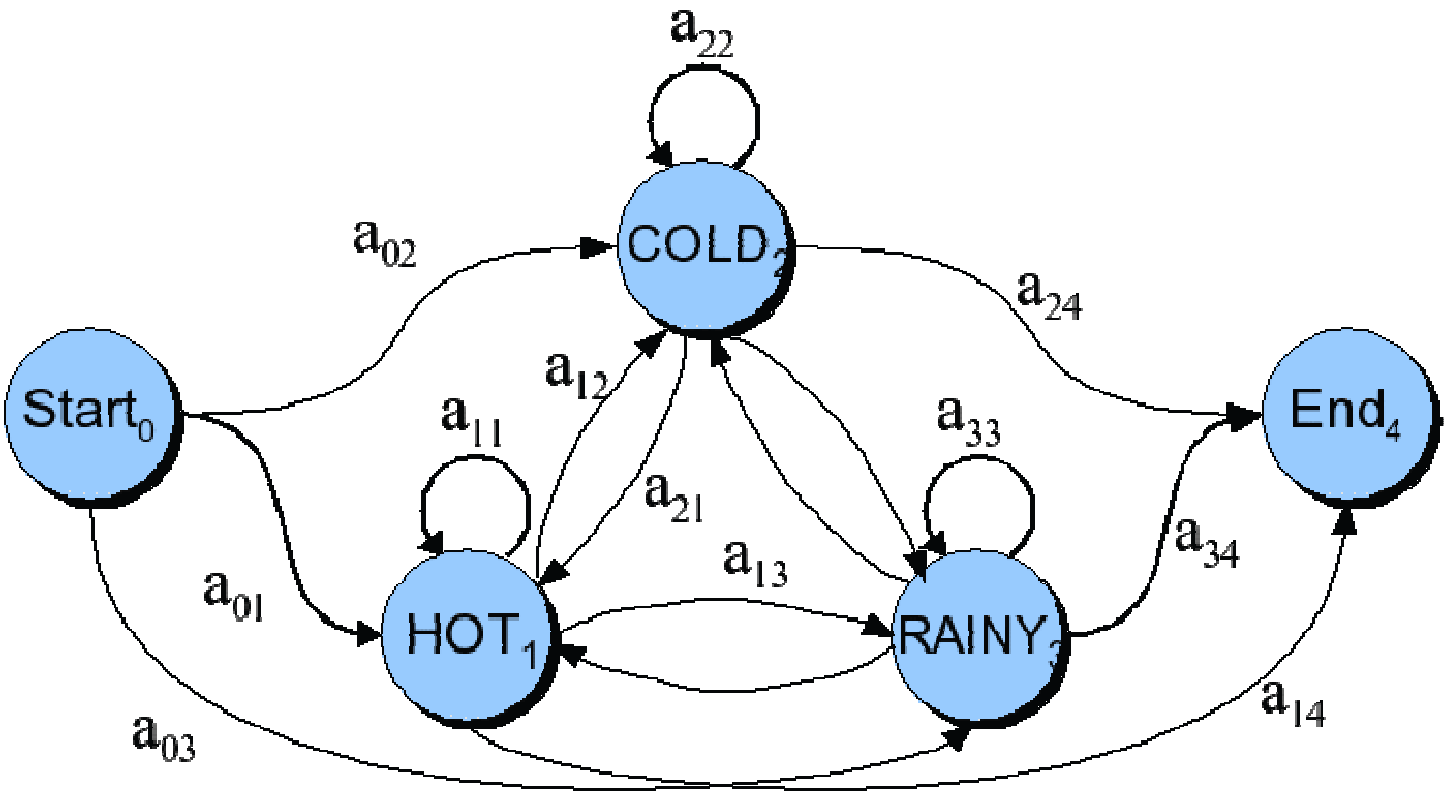
HMM as finite state extension

- A weighted finite-state automaton (WFST) adds probabilities to the arcs between states
  - The sum of the probabilities leaving any arc must sum to one

- A Markov chain is a special case of a WFST in which the input sequence uniquely determines which states the automaton will go through

  uniek: hoogste kans

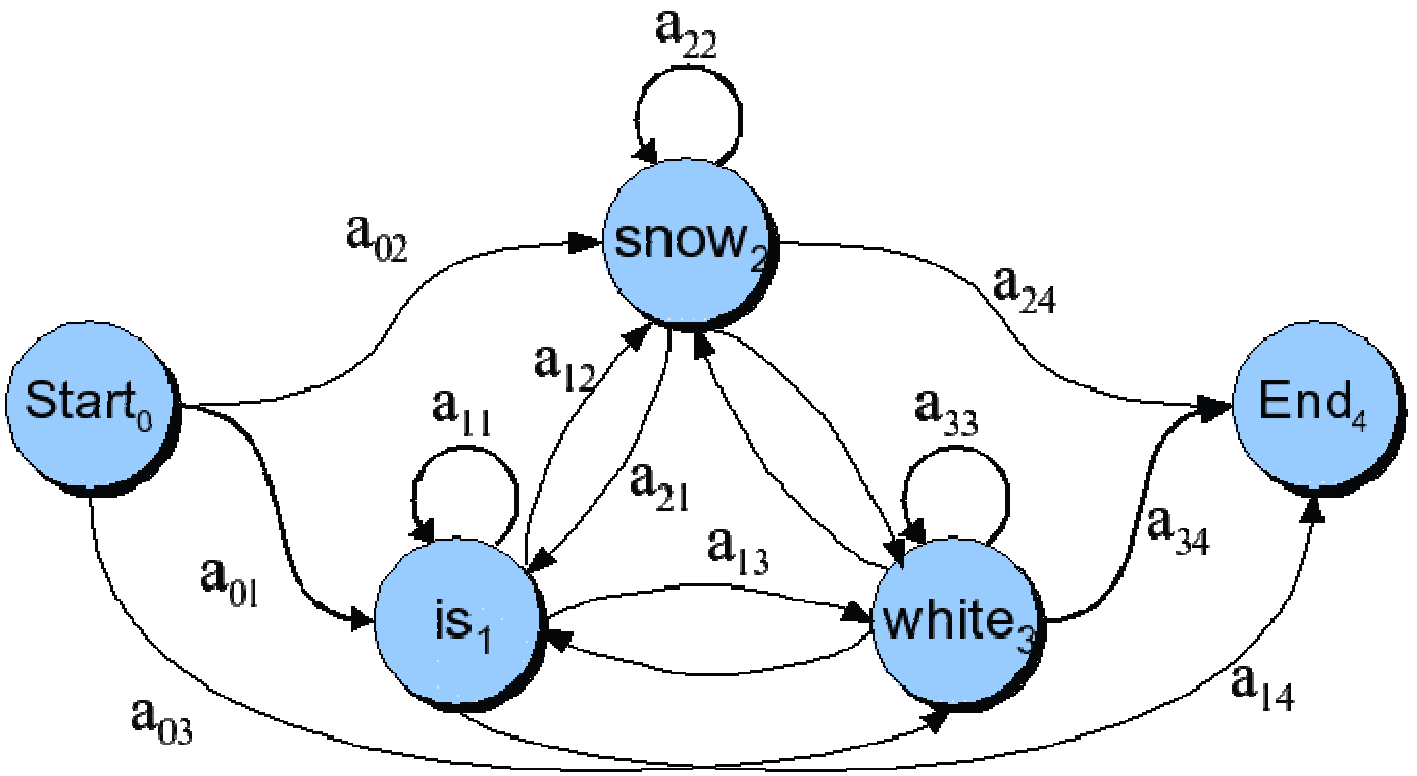
Simpel Markov Chain for Weather



a_{ij} are the transition probabilities between states

start>cold>cold>rainy>hot>rainy>…>*end*

Markov Chain for Words



Start en **End** zijn non-emitting states (leveren geen output)
States 1-3 leveren slechts 1 woord (dus niets is verborgen hier); het voorbeeld heeft geen expliciete tags

Observaties

- Woorden zijn 'observaties' die we kunnen waarnemen: $O= o_1, o_2 \ldots o_N$
- Tags kunnen we **niet** waarnemen (ze staan niet in de tekst), maar ze kunnen wel woorden produceren

- POS: Een state is geassocieerd met een tag, en kan op basis daarvan woorden genereren, die waargenomen kunnen worden
- ASH: Een state is geassocieerd met een foneem, en kan op basis daarvan spraakgeluid produceren, dat we kunnen waarnemen

State definitions

- A set of states (POS: tags)
  - Q = q_1, q_2…q_N
    "state$_t$ *(state at 'position/time' t)*
    is q_i (i kan 1 tot N zijn)"
- Transition probabilities:
  - a set of transition probabilities A = a_{01}, a_{02}..,a_{n1}..,a_{nn}.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition probability matrix A
- Current state only depends on previous state (=eerste orde Markov keten) $P(q_i \mid q_1 .. q_{i-1}) = P(q_i \mid q_{i-1})$

Matrix

- A-matrix (transitiewaarschijnlijkheden)

$$
\begin{array}{cccccc}
a_{01} & a_{02} & a_{03} & a_{04} & a_{05} & a_{0f} \\
a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{1f} \\
a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{2f} \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{3f} \\
a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{4f} \\
a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{5f}
\end{array}
$$

Markov Chain for Weather

- What is the probability of 4 consecutive rainy days?
- Sequence is rainy-rainy-rainy-rainy
- I.e., state sequence is 3-3-3-3
- P(3,3,3,3) =
  - $a_{03}a_{33}a_{33}a_{33} = 0.2 \times (0.6)^3 = 0.06912$

HMM for Ice Cream

- You are a climatologist in the year 2799
- Studying global warming
- You can't find any records of the weather in Baltimore, MA for summer of 2007
- But you find Jason Eisner's diary
- Which lists how many ice-creams Jason ate every date that summer
- Our job: figure out how hot it was

Hidden Markov Model

- For simpel Markov chains, the output symbols are the same as the states.
  - Observe **hot** weather: we're in state **hot**
- But in part-of-speech tagging (and other things)
  - The output symbols are **words**
  - But the hidden states are **part-of-speech tags**
- So we need an extension!

Hidden Markov Model

- A Hidden Markov Model is an extension of a Markov chain in which the input symbols are not the same as the states.

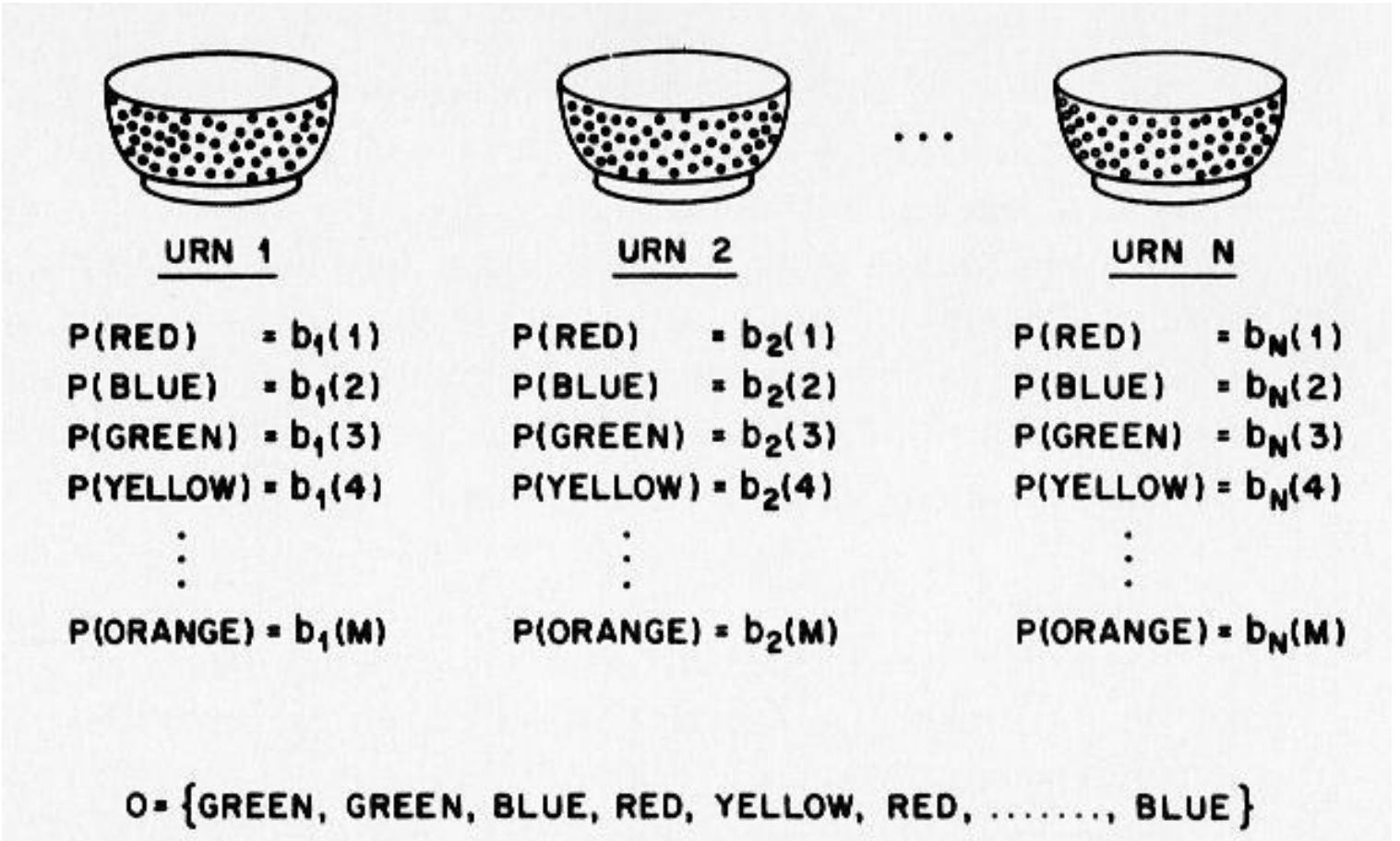
- This means we don't know which state we are in

  when we do some observation,

  since more than one state could have generated the observation

  Nound > walk
  Verb 1sg pres > walk

Urn and balls example (Rabiner)



Hidden Markov Models

- States set Q = q_1, q_2…q_N
  (POS verzameling)
- Observations O= o_1, o_2…o_T
  (een zin die uit woorden bestaat)
- Transition probabilities between states
  (kans op overgang tussen twee POS)
- Observation likelihoods (new!)
  (kans op woord, gegeven de POS)

Hidden Markov Models (formally)

- States set Q = $q_1, q_2 \ldots q_N$
  + start state: q_0, final state: q_F (not emitting)

- Observations O= $o_1, o_2 \ldots o_T$ (observed one after another)
  - Each observation is a symbol from a vocabulary V = $\{v_1, v_2, \ldots v_V\}$

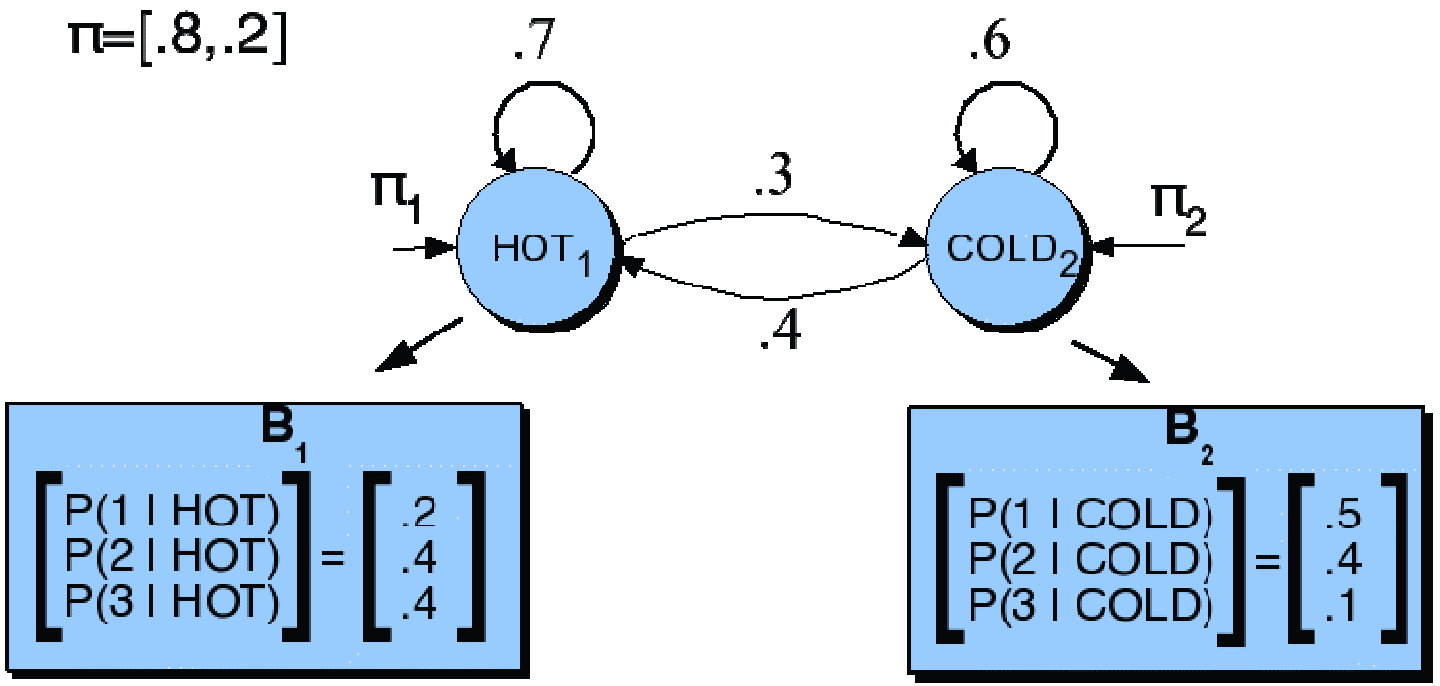
Hidden Markov Models (formally)

- Transition probabilities between states
  - Transition probability matrix $A = \{a_{ij}\}$
    $a_{ij} = P(state_t = q_j | state_{t-1} = q_i)$
- Observation likelihoods (NEW!)
  - Output probability matrix $B = \{b_i(o_t)\}$
    $b_i(o_t) = P(o_t | state_t = q_i)$

Eisner Task

- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,…
    (aantal gegeten ijsjes per dag)
- Produce: (de kans op)
  - Weather Sequence: Hot,Cold,H,H,H,C…

HMM for Ice Cream



Dit is een ***gegeven*** model: de waarden van de kansen ***zijn al bekend***

$\pi_i = a_{0i}$ (de kans om in een state te starten) is andere notatie

Ice cream task

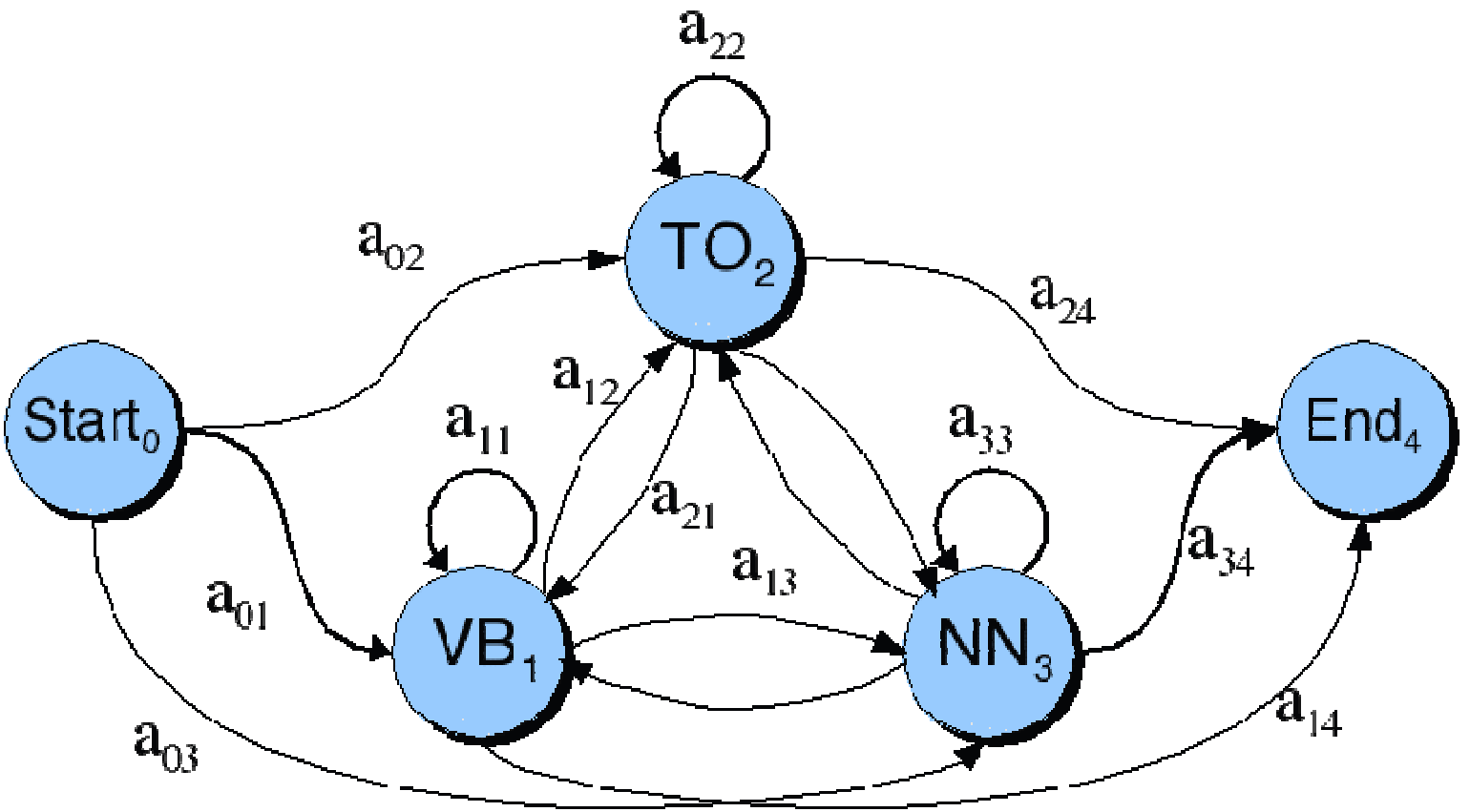
Gegeten aantal ijsjes: 1,2,3,2,2,2

Wat is de kans op de weersequentie: HCHHHC

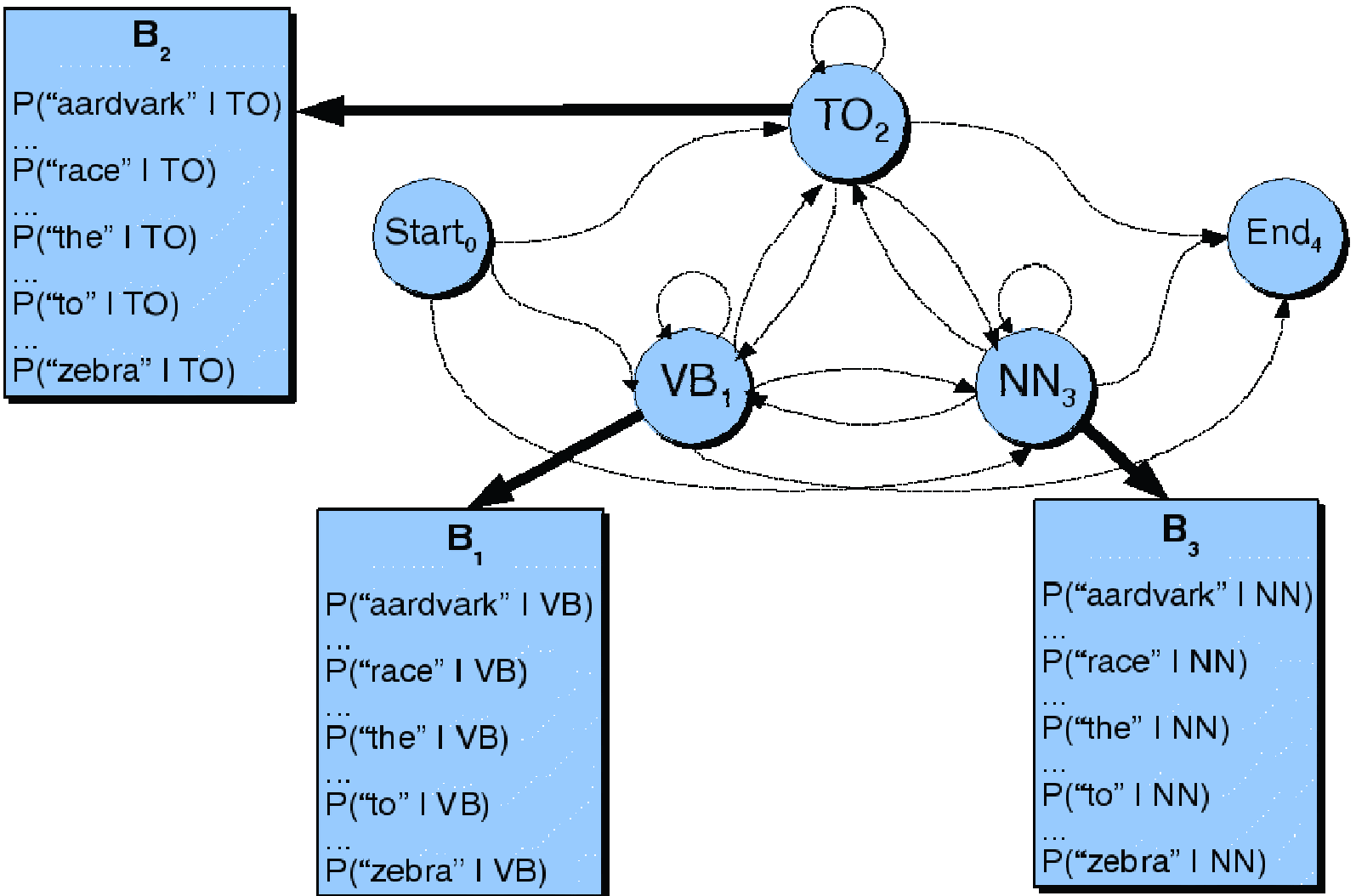
P(HCHHHC) = P(*start* in state hot) *
P(1 ijsje in state hot) *
P(overgang van state hot naar state cold) *
P(2 ijsjes in state cold) *
P(overgang van state cold naar state hot) *
P(3 ijsjes in state hot) *
P(overgang van state hot naar state hot) *
P(2 ijsjes in state hot) *
P(overgang van state hot naar state hot) *
P(2 ijsjes in state hot) *
P(overgang van state hot naar state cold) *
P(2 ijsjes in state cold)

- *Al die kansen zijn bekend (geleerd in een trainingsfase)*

Transition Probabilities (POS states)



Observation Likelihoods



Decoding

- Ok, now we have a complete model that can give us what we need. Recall that we need to get

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

We moeten dus

- *alle mogelijke paden* door het model doorlopen,
- elke keer de kans uitrekenen dat via dat pad de observaties kunnen worden verkregen
- het pad kiezen, dat de hoogste kans oplevert

Dat is veel werk! > Viterbi algoritme

Viterbi

- Viterbi algoritme is een vorm van *dynamisch programmeren (DP)*

- wordt ook gebruikt bij
  - minimal edit distance (spell checkers)
    - spelfout: wat is het meest waarschijnlijke juiste woord?
  - vergelijken van twee gesproken woorden
    - op welke van bekende woordrealisaties, lijkt een onbekend woord het meest?
      Dynamic Time Warping (DTW)
      (komen we straks nog op terug)

Viterbi

- Altijd uitrekenen:
  - De kans dat je in een state (POS) komt
  - De kans dat die state (POS) de observatie (woord) kan leveren die op dat moment in de sequentie (zin) nodig is
  - Kansen vermenigvuldigen: $a_{ij} * b_j(o_t)$
- Start:
  - In theorie mogelijk in alle POS. Bereken:
  - Kans op POS1 als start * de kans dat de POS1 het eerste woord kan leveren = $Viterbi_1(POS1)$

Viterbi

- 1e woord gehad, nu naar het 2e woord
- Onderzoek voor *elke* POS2 hoe die vanuit *elke* POS1 bereikt kan worden: bereken voor alle POS1
  $Viterbi_1(POS1)$ *
  overgangskans POS1 naar POS2 *
  generatiekans op het 2e woord in POS2
- Onthoud voor elke POS2, alleen die POS1 die de maximale totale kans oplevert: $Viterbi_2(POS2)$, onthoud daarbij de betreffende overgang vanaf POS1
- Doe dit voor alle volgende woorden
- Eind: kies de $POS_{eind,max}$ waarvoor de maximale totale kans is verkregen, en reconstrueer het pad daar naartoe van eind tot begin (backtrack)

I want to race

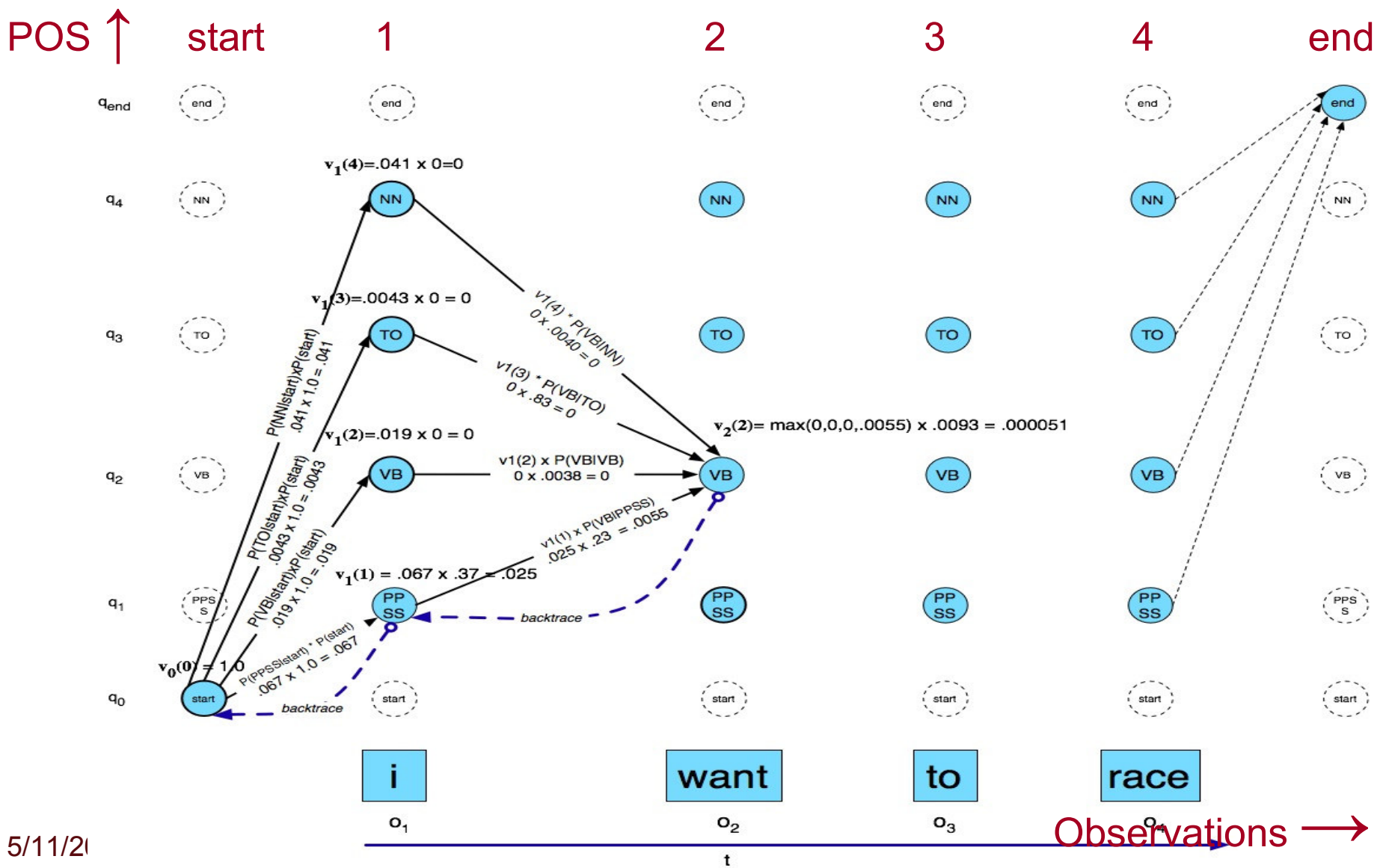
	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

POS transitiematrix

	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

Output waarschijnlijkheden per POS

Viterbi Example



Aantal berekeningen

- 4 states en 4 woorden
  - 4x4x4x4 paden = 256 paden
  - elk pad 4 berekeningen = 1024 berekeningen
- Viterbi
  - 4 x (4x4) berekening = 64 berekeningen
    - Heel efficiënt,
    - levert alleen het beste pad op,
    - maar andere paden zijn (meestal) veel minder kansrijk

The Viterbi Algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*

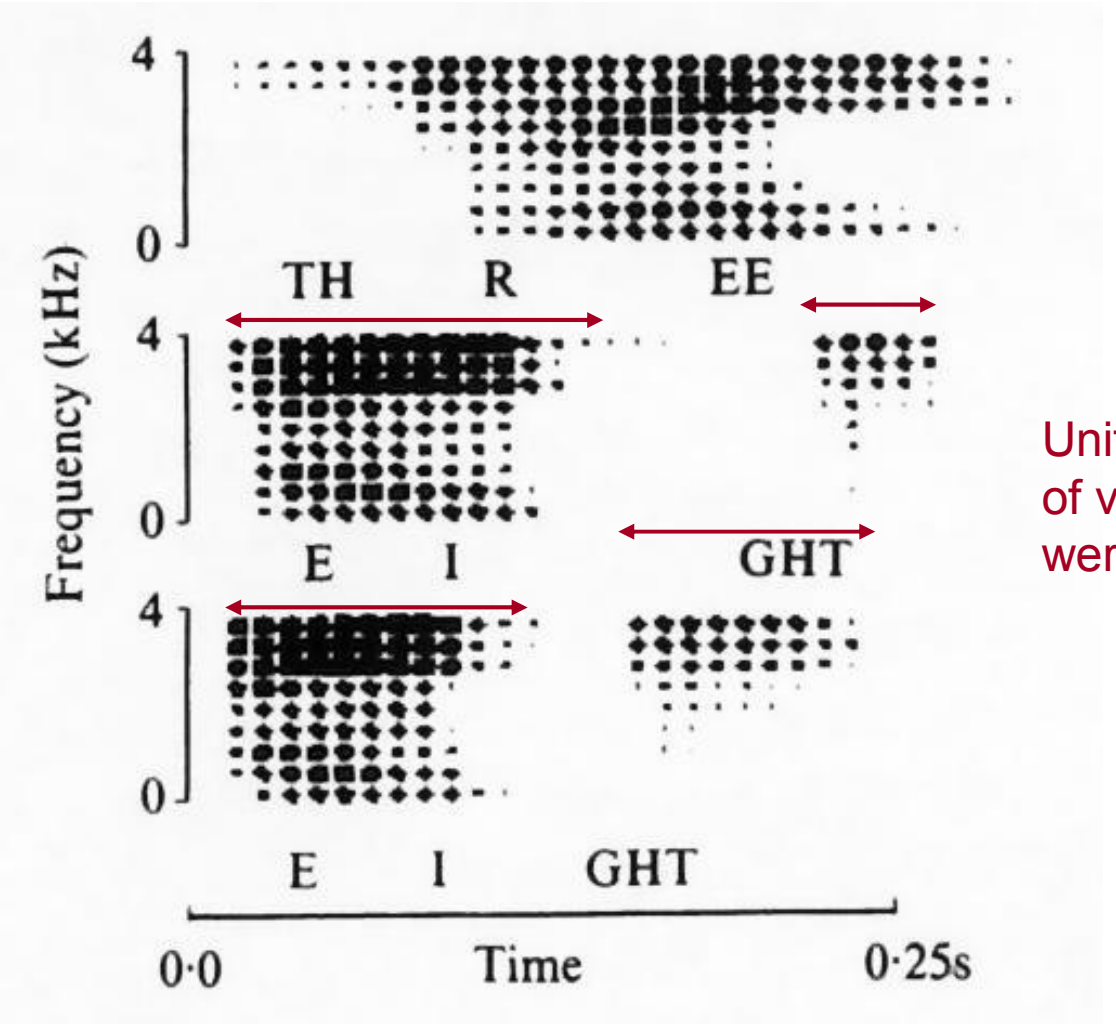
create a path probability matrix *viterbi*$[N+2,T]$
for each state s **from** 1 **to** N **do** ; initialization step
$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**
$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$$
$$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F} \quad \text{; termination step}$$
$$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F} \quad \text{; termination step}$$
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Vergelijkbare pad problemen

- In Spraak: Dynamic Time Warping (DTW)

- Probleem: de akoestische (spectrale) overeenkomst tussen twee woorden uitrekenen.
  - voor automatische spraakherkenning, beschikbaar:
    voorbeeldwoorden,
    en een onbekend woord
  - vraag: welk voorbeeldwoord lijkt het meest op het onbekende woord?
    - hoe kan voor duurverschillen worden gecorrigeerd?

DTW: spectrogrammen

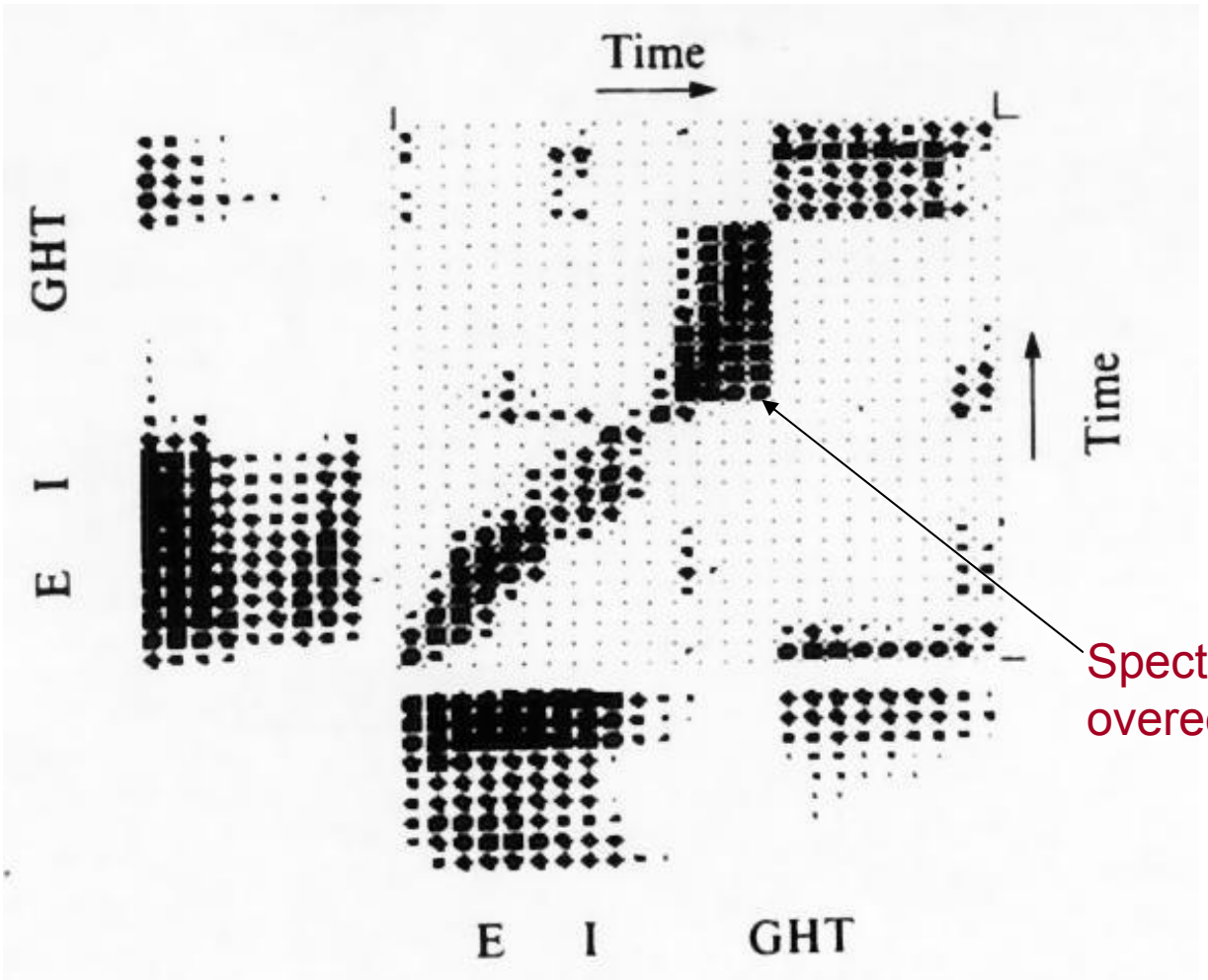


Uniforme verkorting of verlenging werkt niet

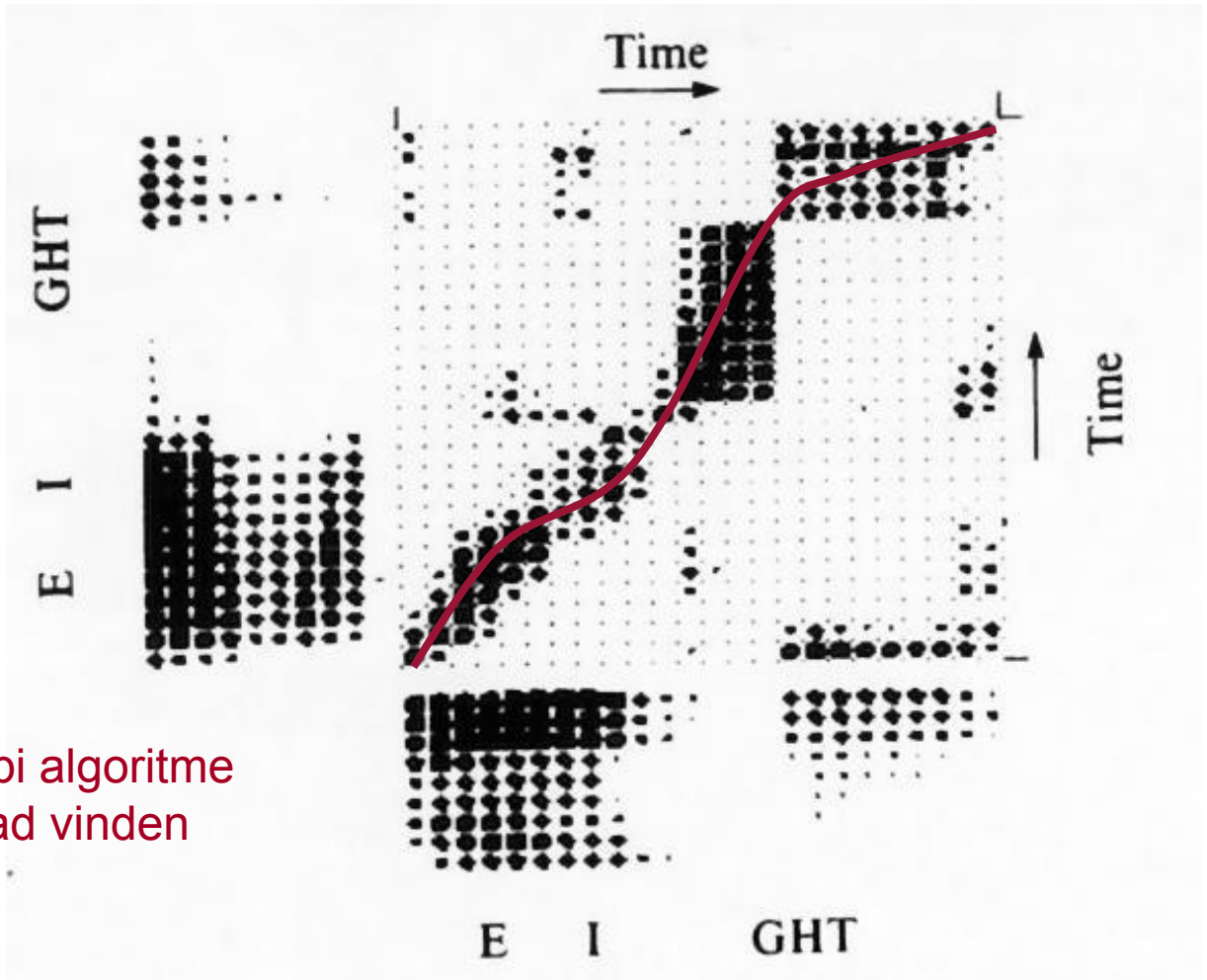
dtw: oplijnen

- Middelste “eight” vergelijken met “three” en andere versie van “eight”

DTW: vergelijk twee woorden op alle tijdstippen



DTW: vind pad dat een maximale spectrale overeenkomst oplevert



Het Viterbi algoritme kan dit pad vinden

Evaluation

- So once you have you POS tagger running how do you evaluate it?
  - Overall error rate with respect to a gold-standard test set.
  - Error rates on particular tags
  - Error rates on particular words
  - Tag confusions...

Error Analysis

- Look at a confusion matrix

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	**3.3**	2.1	1.7	.2	**2.7**
NN		**8.7**	—				.2
NNP	.2	**3.3**	**4.1**	—	.2		
RB	**2.2**	2.0	.5		—		
VBD		.3	.5			—	**4.4**
VBN		**2.8**				**2.6**	—

- See what errors are causing problems
  - Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
  - Preterite (VBD, simpel past) vs Participle (VBN) vs Adjective (JJ)

Evaluation

- The result is compared with a manually coded "Gold Standard"
  - Typically accuracy reaches 96-97%
  - This may be compared with result for a baseline tagger (one that uses no context).
- Important: 100% is impossible even for human annotators.

Summary

- Parts of speech
- Tagsets
- Part of speech tagging
- HMM Tagging
  - Markov Chains
  - Hidden Markov Models